Programming Intentional Agents: Exercises in *Jason*

Autonomous Systems
Sistemi Autonomi

Stefano Mariani, Andrea Omicini
{s.mariani, andrea.omicini}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

Academic Year 2015/2016

1. Getting Started
2. Basic Examples
3. AgentSpeak(L) Example
4. *Jason* Example: Domestic Robot
5. *Jason* Example: ContractNet Protocol

Outline

1. Getting Started
2. Basic Examples
3. AgentSpeak(L) Example
4. *Jason* Example: Domestic Robot
5. *Jason* Example: ContractNet Protocol

Get *Jason*

- go to *Jason* home page

  `http://jason.sourceforge.net/wp/`

- on the right, click on *"This will take you to the SourceForge download page ..."* under *Jason* banner to go to *Jason* download page, on SourceForge

  `http://sourceforge.net/projects/jason/files/`

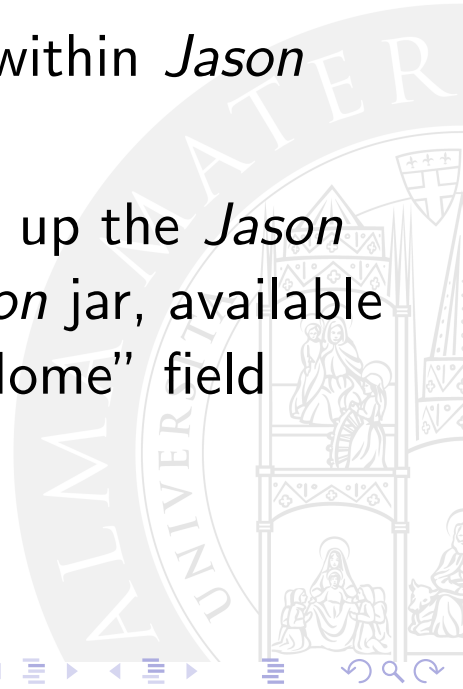
- click on the quick link next to *"Looking for the latest version?"* to obtain *Jason* latest version (currently, **1.4.2**)

Install *Jason* I

- *Jason* comes with its own IDE (jEdit enhanced with *Jason* plugin), but also a Eclipse plugin exists: we will use Eclipse
- once downloaded *Jason* bundle, unpack it in any directory then run

  ```
  java -jar lib/jason.jar
  ```

  in a command prompt—assuming you are positioned within *Jason* directory, e.g. `Jason-1.4.2/`
- a configuration window should pop-up, letting you set up the *Jason* runtime environment properties—e.g., location of *Jason* jar, available distribution infrastructures, etc.): be sure the "Java Home" field points to the JVM you want to use



Install *Jason* II

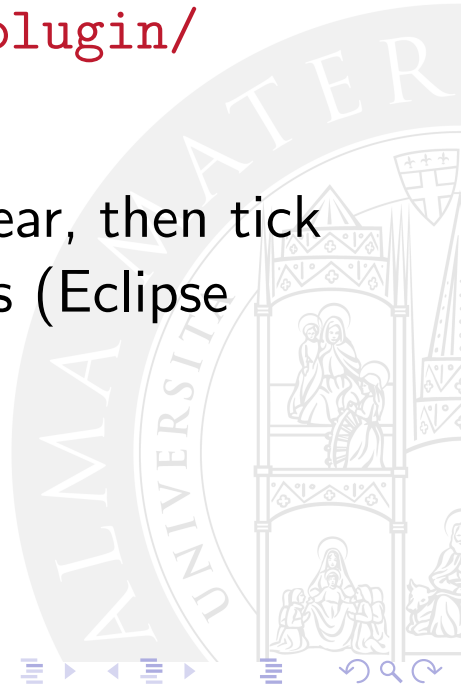
- now open Eclipse and click "Help > Install New Software. . . ' > Add.."', then type in the "Location" field

  `http://jason.sourceforge.net/eclipseplugin/juno/`

  for Juno or newer

  `http://jason.sourceforge.net/eclipseplugin/`

  for Indigo
- click "Ok" and wait for the "jasonide" feature to appear, then tick the checkbox and step through the installation process (Eclipse restart included)



Outline

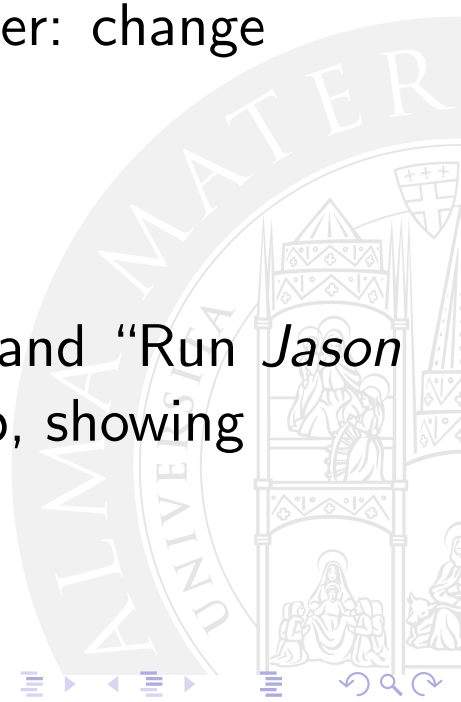
1. Getting Started
2. Basic Examples
3. AgentSpeak(L) Example
4. *Jason* Example: Domestic Robot
5. *Jason* Example: ContractNet Protocol

Hello World

- in Eclipse, open the Jason perspective ("Window > Open Perspective > Other... > *Jason*")
- then create a new *Jason* project (click on "New" icon on top left corner then selects "*Jason* Project")
- name the project and leave default options (centralised infrastructure without environment)
- two files are automatically created (filenames *do* matter: change them wisely)

  .asl is the *Jason* agent source file
  .mas2j is the *Jason* MAS configuration file

- give a look at the code then right-click on .mas2j file and "Run *Jason* Application": the MAS console window should pop-up, showing `agent1` printing "hello world."
- click button "! Stop" to stop MAS execution



Greeting

- go to

  `http://jason.sourceforge.net/mini-tutorial/getting-started/`

- go to section "Creation of a simple example" and follow steps 1-8 (ignore JADE distribution) to set up another simple example[1]
- create a new project as done for the "hello world" example
- create new agents by right-clicking on `src/asl/` source folder then selecting "New > Agent" and giving them a name (leave other fields as default): this will automatically add the newly created agent to the .mas2j configuration file

[1] Icons do not match because the example is done in jEdit, not Eclipse.

Outline

1 Getting Started

2 Basic Examples

3 AgentSpeak(L) Example

4 *Jason* Example: Domestic Robot

5 *Jason* Example: ContractNet Protocol

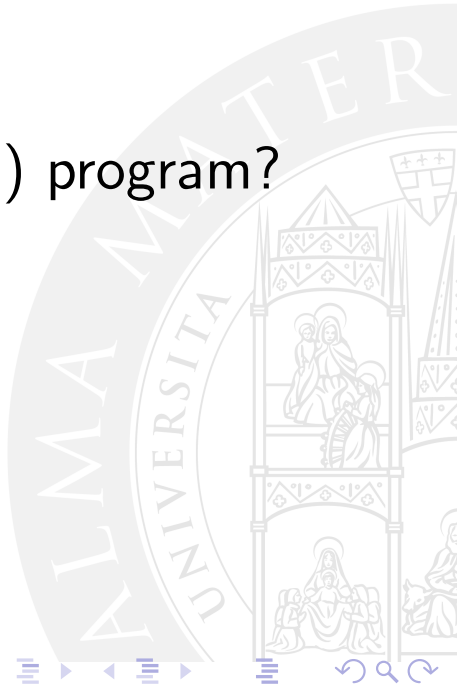
Import *Jason* Project

- go to this lab APICe page[2] and download `sa1516_agentSpeakL.zip`
- unzip it anywhere you want
- in Eclipse click "File > Import... > *Jason*/*Jason* Project" and in "Select root directory" browse to the folder you just unzipped (the one storing the .mas2j file), then click "Finish" (check "Copy projects into workspace" to have the source files copied, not linked)

[2] `http://apice.unibo.it/xwiki/bin/view/Courses/Sa1516Lab`

AgentSpeak(L)

- Very basic example of AgentSpeak(L) program featuring
  - achievement-goal addition events
  - belief addition events
  - plan contexts
  - test-goals
- Can you spot what should *not* be in an AgentSpeak(L) program?
  Hint: but, it *can* be in a *Jason* program...



Outline

1 Getting Started

2 Basic Examples

3 AgentSpeak(L) Example

4 *Jason* Example: Domestic Robot

5 *Jason* Example: ContractNet Protocol

Scenario

- go to this course APICe page[3] and download `sa1516_domesticRobot.zip` from the attachments

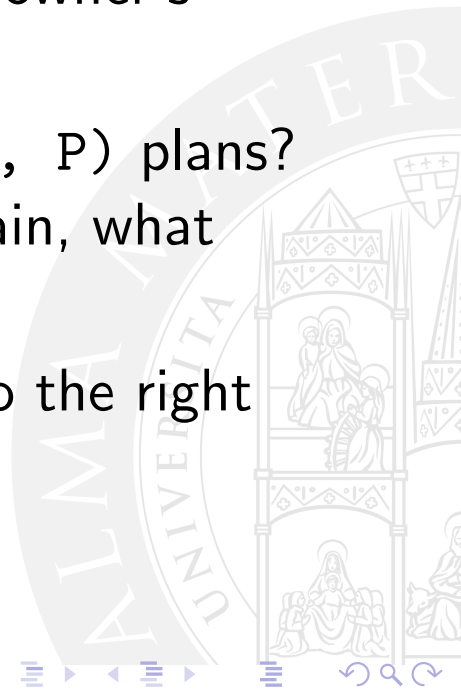
Domestic Robot

- A domestic robot has the goal of serving beer to its owner
- Thus, it receives beer requests from the owner, goes to the fridge, takes out a beer, brings it back to the owner
- The robot eventually orders beer using a nearby supermarket's home delivery service
- Also, the robot obeys hard-wired rules from the Department of Health (e.g. "do not serve more than 10 beers a day")

[3] `http://apice.unibo.it/xwiki/bin/view/Courses/Sa1516Lab`

Highlights

- The robot should remember if beer is available irrespective of its location in the environment, because the perception about beer stock is available *only when the robot is in front of the open fridge*—as soon as it closes the fridge, the perception is gone
- How to ensure that the robot will respond *only* to its owner's requests?
- What happens if we change the order of `+!at(robot, P)` plans? And if we drop the *plan context* from either one? Again, what happens if we change plans order in this case?
- How to ensure each external action is available only to the right agent?



Outline

1 Getting Started

2 Basic Examples

3 AgentSpeak(L) Example

4 *Jason* Example: Domestic Robot

5 *Jason* Example: ContractNet Protocol

Scenario

- go to this course APICe page[4] and download `sa1516_contractNetProtocol.zip` from the attachments

ContractNet Protocol

- One agent, called "initiator", wishes to have some tasks performed, thus asks other agents, called "participants", to bid to perform that task
- This asking message is called "call for proposals" (cfp, for short), and participants may reply by either sending their proposals or by refusing the call
- It can happen that participants do not even reply, so when a deadline chosen by the initiator expires, it evaluates the received proposals and selects one agent to perform the task

[4] `http://apice.unibo.it/xwiki/bin/view/Courses/Sa1516Lab`

Highlights

- Note that plan `@contracting` is `atomic`: when it starts executing, *no other intention is selected for execution* before it finishes:
  - the first action of the plan is to change the protocol state, which is also used in the context of the plan
  - this ensures that the intention for the goal `!contract` is never performed twice
- Suppose the initiator wants to cancel the cfp
  - add a plan in the initiator program for events such as `+!abort(CNPId)`
  - which kind of *illocutionary force* (or performative) may this plan exploit to inform participants accordingly?

hint basically, we want to remove an agent's belief from another agent...

1. Getting Started
2. Basic Examples
3. AgentSpeak(L) Example
4. *Jason* Example: Domestic Robot
5. *Jason* Example: ContractNet Protocol

Programming Intentional Agents: Exercises in *Jason*

Autonomous Systems
Sistemi Autonomi

Stefano Mariani, Andrea Omicini
{s.mariani, andrea.omicini}@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

Academic Year 2015/2016